Logic Error Localization and Correction with Machine Learning (Student Abstract)

Zhenyu Xu[1], **Keyi Lu**[2], **Victor S.Sheng**[1]

[1] Department of Computer Science, Texas Tech University
[2] Department of Computer Science and Engineering, The Ohio State University
zhenxu@ttu.edu, lu.2285@osu.edu, victor.sheng@ttu.edu

Abstract

We aim to propose a system repairing programs with logic errors to be functionally correct among different programming languages. Logic error program repair has always been a thorny problem: First, a logic error is usually harder to repair than a syntax error in a program because it has no diagnostic feedback from compilers. Second, it requires inferring in different ranges (i.e., the distance of related code lines) and tracking symbols across its pseudocode, source code, and test cases. Third, the logic error datasets are scarce, since an ideal logic error dataset should contain lots of components during the development procedure of a program, including a program specification, pseudocode, source code, test cases, and test reports (i.e., test case failure report). In our work, we propose novel solutions to these challenges. First, we introduce pseudocode information to assist logic error localization and correction. We construct a code-pseudocode graph to connect symbols across a source code and its pseudocode and then apply a graph neural network to localize and correct logic errors. Second, we collect logic errors generated in the process of syntax error repairing via DrRepair from 500 programs in the SPoC dataset and reconstruct them to our single logic error dataset, which we leverage to train and evaluate our models. Our experimental results show that we achieve 99.39% localization accuracy and 19.20% full repair accuracy on logic errors with five-fold cross-validation. Based on our current work, we will replenish and construct more complete public logic error datasets and propose a novel system to comprehend different programming languages from several perspectives and correct logic errors to be functionally correct.

Introduction

Logic errors and syntax errors are both types of errors that threaten the program life cycle, but we found that most research on program repair ignores logic error correction. Automated Program Repair (APR) is an area of research focused on the automatic generation of bug-fixing patches. Most recent work focuses on localizing and correcting syntax errors. They feed abstract features of the code to a neural network for classification. Gupta et al. (Gupta et al. 2017) proposed a seq2seq machine learning approach for repairing syntax errors in C. Yasunaga et al. (Yasunaga et al. 2020) focus on the problem of syntax error repair and design a program-feedback graph constructed from source codes and compiler feedback. Logic errors cannot be caught by program compilers. A logic error does not cause any abnormal termination or crash in the program running time, but it will cause an incorrect answer or a wrong action with serious consequences. Some common types and descriptions of logic errors are shown in Table 1 (Yoshizawa et al. 2019). Logic errors in a program can be very frustrating for programmers to locate and repair. Current automated program repair (APR) research focuses on syntax error repair and ignores logic errors. The overall goal of this research is to develop an intelligent system for automatically localizing and correcting logic errors.

Type of Logic error	Description
Loop condition	Incorrect loops in the for/while condition
Condition branch	Incorrect expression in the if condition
Statement integrity	Statement lacks a part of logical structure
Output/Input format	Incorrect cin/cout statement
Variable initialization	Incorrect declaration of variables
Data type	Incorrect data type
Computation	Incorrect basic math symbols

Table 1: Types and descriptions of common logic errors

Our Approach

Logic-Err dataset The proposed Logic-Err dataset contains 500 different programs, 871 logic errors (1.74 logic errors per program on average), public test cases, hidden test cases, and pseudocodes for each line. It is constructed based on SPoC programs repaired by DrRepair. DrRepair (Yasunaga et al. 2020) generates a series of repaired programs by iterative replacing a predicted error line with a modified code line. One iteration generates one repaired program. During the repairing process of DrRepair, we can obtain multiple repaired programs with only logic errors, and construct our Logic-Err dataset.

Model Fig. 1(i) illustrates our logic error repairing model architecture. It uses a seq2seq (sequence to sequence) structure with an encoder and decoder. The encoder takes in a source code and its corresponding pseudocode, and

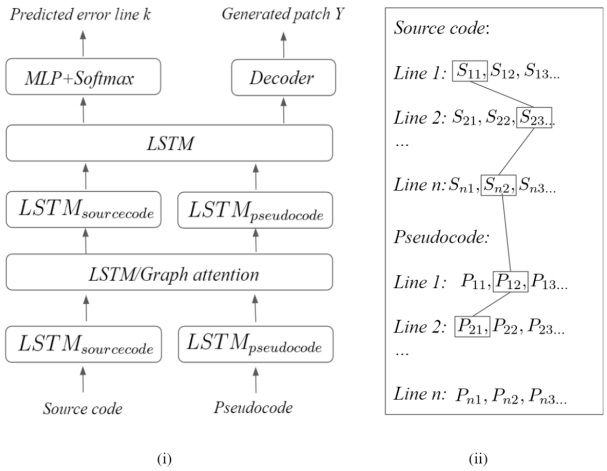

Figure 1: (i) Model architecture as LSTM and LSTM + Graph (ii) Graph edges across a source code and its pseudocode and its generated graph attention layer with multi-heads

the decoder predicts a distribution of erroneous line k over code lines and generates a repaired code line. When this graph attention layer is replaced by a pure LSTM layer, the whole model structure is a pure LSTM model. Otherwise, it is an LSTM+Graph model. A training example consists of source codes S with logic errors, pseudocode P, a logic error line index k, and the repaired line Y_k. In Fig. 1(ii), several nodes are connected (e.g., S_{11}, $S21\ldots$). Each node will be assigned an average of weighted attention over its neighboring nodes. A weighted hidden state h_i is calculated using an aggregation function in the graph attention layer as defined in Equation 1. It will allocate different weights to different edges with attention coefficients. The graph structure allows token tracing efficiently, and graph attention mechanisms help focus on important tokens. We introduce a graph attention mechanism in graph G to give high weights to important tokens. The input of the graph attention layer is a set of token representations. We merge all the information via averaging on the final layer of the attention network.

Experiment and Results

We use two criteria to evaluate the model performance: localization accuracy and repair accuracy, correctly localizing and repairing logic error lines respectively. We conduct preliminary experiments to investigate the logic error correction performance of DrRepair, the LSTM model, and the LSTM+Graph model. As we know, DrRepair reaches a relatively high accuracy in repairing syntax errors, achieving 93.1% localization accuracy and 53.0% repair accuracy on the SPoC syntax error dataset. It is interesting to investigate the performance of DrRepair in terms of logic error correction. The experimental results of the three models are shown in Table 2. From Table 2, we can see that DrRepair does not perform well in terms of localizing and repairing logic errors, and LSTM and LSTM+Graph perform much better than DrRepair. LSTM+Graph performs the best, much better than LSTM. The localization accuracy of LSTM+Graph is 68.17% higher than the pure LSTM model, repair accuracy is 8.1% higher than the pure LSTM model. This suggests that the graph used in the LSTM+Graph model plays a crucial role in logic error localizing and repairing separately.

Type of Models	Localization Accuracy	Repair Accuracy
DrRepair(base+graph)	0.21%	0.00%
LSTM(code only)	31.22%	11.10%
LSTM+Graph(edges across code and pseudocode)	99.39%	19.20%

Note: DrRepair is trained on the SPoC dataset and evaluated on the Logic-Err dataset; both LSTM and LSTM+Graph are trained and evaluated on the Logic-Err dataset through 5-fold cross-validation

Table 2: We evaluate the performance of different model types on the Logic-Err dataset (871 programs) with five-fold cross-validation

Conclusion

We introduced a novel graph-based neural network to localize and repair logic errors. Our graph-based model takes a novel perspective on the logic error correction problem and adopts pseudocode to construct an efficient graph structure, which helps to localize logic error accurately and use the seq2seq model structure to predict logic error lines and generate patches. Our proposed model LSTM+Graph improves the accuracy of localization and repair of logic errors effectively.

References

Ettles, A.; Luxton-Reilly, A.; and Denny, P. 2018. Common logic errors made by novice programmers. In *Proceedings of the 20th Australasian Computing Education Conference*, 83–89.

Gupta, R.; Pal, S.; Kanade, A.; and Shevade, S. 2017. Deepfix: Fixing common c language errors by deep learning. In *Thirty-First AAAI conference on artificial intelligence*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Yasunaga, M.; and Liang, P. 2020. Graph-based, self-supervised program repair from diagnostic feedback. In *International Conference on Machine Learning*, 10799–10808. PMLR.

Yoshizawa, Y.; and Watanobe, Y. 2019. Logic error detection system based on structure pattern and error degree. *Advances in Science, Technology and Engineering Systems Journal*, 4(5): 1–15.